Zend_Service_Instagram

<ac:macro ac:name="unmigrated-inline-wiki-markup"><ac:plain-text-body><![CDATA[

<ac:macro ac:name="unmigrated-inline-wiki-markup"><ac:plain-text-body><![CDATA[

Zend Framework: Zend_Service_Instagram Component Proposal

Proposed Component Name	Zend_Service_Instagram
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Service_Instagram
Proposers	Mauricio Cuenca
Zend Liaison	TBD
Revision	1.0 - 06 March 2011: Initial Draft. (wiki revision: 21)

Table of Contents

1. Overview
2. References
3. Component Requirements, Constraints, and Acceptance Criteria
4. Dependencies on Other Framework Components
5. Theory of Operation
6. Milestones / Tasks
7. Class Index
8. Use Cases
9. Class Skeletons

1. Overview

Zend_Service_Instagram is a full implementation of the official Instagram API. It provides an object oriented interface to the API and returns data in JSON format or PHP data objects.

2. References

- Instagram Developer
- Instagram PHP API implementation

3. Component Requirements, Constraints, and Acceptance Criteria

- This component **will** fully implement Instagram official API.
- This component should namespace the API methods as they are stated in Instagram's API documentation.
- This component **will not** support real-time photo updates.

4. Dependencies on Other Framework Components

- Zend_Oauth
- Zend_Service_Exception
- Zend_Http_Client
- Zend_Http_Client_Adapter_Curl
- Zend_Http_Client_Exception

5. Theory of Operation

Zend_Service_Instagram will allow a developer an easy interaction with the Instagram API in an object oriented manner, supporting such actions as getting users' information, browse through popular media, manage users' relationships, comments, likes, tags and locations. All data returned can be represented whether in JSON format or as PHP data objects. Zend_Service_Instagram creates a PHP client for the Instagram API.

6. Milestones / Tasks

- Milestone 1: Proposal, Community Review and Acceptance.
- Milestone 2: Unit tests exist, work, and are checked into versioning system.
- Milestone 4: Initial documentation exists.

7. Class Index

- Zend_Instagram_Exception
- Zend_Instagram

8. Use Cases

UC-01 Users

UC-02 Relationships

UC-03 Media

UC-04 Comments

UC-05 Likes

UC-06 Tags

UC-07 Locations

9. Class Skeletons

```
require_once 'Zend/Http/Client.php';

class Zend_Service_Instagram extends Zend_Service {

    /**
     * The name of the GET param that holds the authentication code
     * @var string
     */
    const RESPONSE_CODE_PARAM = 'code';

    /**
     * Format for endpoint URL requests
     * @var string
     */
    protected $_endpointUrls = array(
        'user' => 'https://api.instagram.com/v1/users/%d/?access_token=%s',
        'user_search' => 'https://api.instagram.com/v1/users/search?q=%s&access_token=%s',
        'user_follows' => 'https://api.instagram.com/v1/users/%d/follows?access_token=%s',
        'media' => 'https://api.instagram.com/v1/media/%d?access_token=%s',
        'media_popular' => 'https://api.instagram.com/v1/media/popular?access_token=%s',
        'media_comments' => 'https://api.instagram.com/v1/media/%d/comments?access_token=%s',
        'post_like' => 'https://api.instagram.com/v1/media/%d/likes,',
        'remove_like' => 'https://api.instagram.com/v1/media/%d/likes?access_token=%s',
        'tags' => 'https://api.instagram.com/v1/tags/%s?access_token=%s',
        'tags_search' => 'https://api.instagram.com/v1/tags/search?q=%s&access_token=%s',
        'locations' => 'https://api.instagram.com/v1/locations/%d?access_token=%s',
    );

    /**
    * Configuration parameter
    */
    protected $_config = array();

    /**
     * Access token
     * @var string
     */
    protected $_accessToken = null;

    /**
     * Holds the HTTP client instance
     * @param Zend_Http_Client $httpClient
     */
    protected $_httpClient = null;

    /**
     * Constructor needs to receive the config as an array
     * @param mixed $config
     */
    public function __construct($config = null) {
        $this->_config = $config;
        if (empty($config)) {
            throw new InstagramException('Configuration params are empty or not an array.');
        }
    }

    /**
     * The init method is triggered on every call to retrieve the
     * access token in case is not in the current instance.
     */
    protected function _init() {}
```

```
    /**
     * Gets the code param received during the authorization step
     */
    public function getCode() {
        return $_GET[self::RESPONSE_CODE_PARAM];
    }

    /**
     * Retrieves the OAuth token to be used in every request
     * @return string
     */
    protected function _getOauthToken() {}

     /**
      * Get basic information about a user.
      * @param $id
      * @param $oauthToken
      */
    public function getUser($id) {}

    /**
     * See the authenticated user's feed.
     * @param integer $maxId. Return media after this maxId.
     * @param integer $minId. Return media before this minId.
     */
    public function getUserFeed($maxId = null, $minId = null) {}

    /**
     * Get the most recent media published by a user.
     * @param $id. User id
     * @param $maxId. Return media after this maxId
     * @param $minId. Return media before this minId
     * @param $maxTimestamp. Return media before this UNIX timestamp
     * @param $minTimestamp. Return media after this UNIX timestamp
     */
    public function getUserRecent($id, $maxId = '', $minId = '',
        $maxTimestamp = '', $minTimestamp = '') {}

    /**
     * Search for a user by name.
     * @param string $name. A query string
     */
    public function searchUser($name) {
    }

    /**
     * Get the list of users this user follows.
     * @param integer $id. The user id
     */
    public function getUserFollows($id) {}

    /**
     * Get the list of users this user is followed by.
     * @param integer $id
     */
    public function getUserFollowedBy($id) {}

    /**
     * List the users who have requested this user's permission to follow
     */
    public function getUserRequestedBy() {}

    /**
     * Get information about the current user's relationship
     * @param integer $id
     */
    public function getUserRelationship($id) {}

    /**
     * Modify the relationship between the current user and the target user
```

```
     * In order to perform this action the scope must be set to 'relationships'
     * @param integer $id
     * @param string $action. One of follow/unfollow/block/unblock/approve/deny
     */
    public function modifyUserRelationship($id, $action) {}

    /**
     * Get information about a media object.
     * @param integer $mediaId
     */
    public function getMedia($id) {}

    /**
     * Search for media in a given area.
     * @param float $lat
     * @param float $lng
     * @param integer $maxTimestamp
     * @param integer $minTimestamp
     * @param integer $distance
     */
    public function mediaSearch($lat, $lng, $maxTimestamp = '',
        $minTimestamp = '', $distance = '') {}

    /**
     * Get a list of what media is most popular at the moment.
     */
    public function getPopularMedia() {}

    /**
     * Get a full list of comments on a media.
     * @param integer $id
     */
    public function getMediaComments($id) {}

    /**
     * Create a comment on a media.
     * @param integer $id
     * @param string $text
     */
    public function postMediaComment($id, $text) {}

    /**
     * Remove a comment either on the authenticated user's media
     * @param integer $mediaId
     * @param integer $commentId
     */
    public function deleteComment($mediaId, $commentId) {}

    /**
     * Get a list of users who have liked this media.
     * @param integer $mediaId
     */
    public function getLikes($mediaId) {}

    /**
     * Set a like on this media by the currently authenticated user.
     * @param integer $mediaId
     */
    public function postLike($mediaId) {}

    /**
     * Remove a like on this media by the currently authenticated user.
     * @param integer $mediaId
     */
    public function removeLike($mediaId) {}

    /**
     * Get information about a tag object.
     * @param string $tagName
     */
```

```
    public function getTags($tagName) {}

    /**
     * Get a list of recently tagged media.
     * @param string $tagName
     * @param integer $maxId
     * @param integer $minId
     */
    public function getRecentTags($tagName, $maxId = '', $minId = '') {}

    /**
     * Search for tags by name - results are ordered first as an exact match
     * @param string $tagName
     */
    public function searchTags($tagName) {}

    /**
     * Get information about a location.
     * @param integer $id
     */
    public function getLocation($id) {}

    /**
     * Get a list of recent media objects from a given location.
     * @param integer $locationId
     */
    public function getLocationRecentMedia($id, $maxId = '', $minId = '',
        $maxTimestamp = '', $minTimestamp = '') {}

    /**
     * Search for a location by name and geographic coordinate.
     * @see http://instagr.am/developer/endpoints/locations/#get_locations_search
     * @param float $lat
     * @param float $lng
     * @param integer $foursquareId
     * @param integer $distance
```

```
     */
    public function searchLocation($lat, $lng, $foursquareId = '', $distance = '') {}
}
```

]]></ac:plain-text-body></ac:macro>
]]></ac:plain-text-body></ac:macro>